TP 5 Data Mining: Standard Neural Network and back propagation

Tuesday 19th May, 2020
deadline: Monday 8th June, 2020, 23:59
OBLIGATORY - INDIVIDUAL WORK

This TP is obligatory, meaning that if you don't submit it you cannot participate to the exam of the course. As for the previous TPs you have to submit both your code and report (you can also use the notebook for your report if you prefer.

Description

In this TP you are going to implement a two-layer fully-connected neural network. The network has an input dimension of N, a hidden layer dimension of H, and performs classification over C classes. We train the network with a softmax loss function. The network uses a Sigmoid activation function after the first fully connected layer.
In other words the architecture:of the network is the following:

input - fully connected layer - Sigmoid - fully connected layer - softmax

Instructions

Before starting open the NN.py and understand the NN class. You have to fill the missing functions of this file.

- Fill the missing parts of the NN.loss(). This function takes the data and weights and computes the class scores, the loss, and the gradients on the parameters.
  - perform the forward pass
  - perform the backward pass
- Fill the missing part in the NN.train()

  - You have just to to update the parameters of the network

- Implement the NN.predict()
  - This method predicts the labels for data points

- Fill the TP5NeuralNet.ipynb jupyter notebook in order to train your network. Train your network for different neurons in the hidden layer (hidden size) and size of mini batch. Set the number of the iteration equal to 500 iteration and add a stopping criterion. Calculate the time it takes to train the network. Find the best hyperparameters by tuning on the validation set.
  - Comment **in details** how the different neurons in the hidden layer (hidden size) and size of mini batch influence the performance of the classifier. Which is the effect of very large/ small batch, hidden size ? How each one change the prediction? Which (and why) is the optimal why to update the weights? etc..

- For the best model 1. compute the accuracy (test accuracy) 2. plot the history of the loss(train) and the training and validation accuracy 3. comment.

- Use $1, 10, 100, 1000$ neurons in the hidden layer (hidden size) and for each case compute the test accuracy and calculate the time it takes to train the network.
  - Explain if and how the size of the hidden layer influences the prediction (validation accuracy) and the computational time to train the network.

- The size of the batch is given as argument in the train() function and in your experiments you will use online SGD, 2. SGD with mini batch = 200, 3. SGD with mini batch = 2000, 4.mini_batch = size of the training data set (Gradient Descent, GD)
  - Explain if and how the batch influences the prediction (validation accuracy) and the computational time to train the network.

Reminders

- The Softmax classifier is the generalization of the binary Logistic Regression classifier to multiple classes. Tthe Softmax classifier gives as output normalized class probabilities

- When we minimize the cost function using Gradient Descent (GD) the weights are updated after seeing all the training instances

- When we minimize the cost function using Stachastic Gradient Descent (SGD) the weights are updated after seeing a mini batch the training instances. When the size of the mini batch is equal to 1 is called single sample update/ online SGD and the weights are updated each time we see a new instance.